DOCUMENT RESUME

ED 083 818 EM 011 541

AUTHOR	Oakman, Robert L.
TITLE	Computer Education for the Humanities: Multiple Possibilities at the University of South Carolina.
PUB DATE	Jun 73
NOTE	4p.; Paper presented at the Conference on Computers in the Undergraduate Curricula (Claremont, California, June 18-20, 1973)
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Computer Assisted Instruction; Computer Oriented Programs; Computer Programs; *Computers; Computer Science; *Computer Science Education; Higher Education; *Humanities; *Humanities Instruction; Liberal Arts Majors; Program Descriptions; *Programing; Programing Languages; Undergraduate Study
IDENTIFIERS	CAI; Non Numerical Processes; University of South Carolina

ABSTRACT

Two sequences of courses at the University of South Carolina offer liberal arts students an introduction to computers and to the relationship between technology and the humanities. These provide training in rigorous thinking and new means of probing the record of human experience. The lower level sequence consists of a general computer appreciation course and an introduction to non-numerical processes. In the latter, students learn about the fundamental concepts of computers and the basics of programing, practice programing, and become involved in large scale computer applications by undertaking a project involving the analysis of a program, the design of an input data structure, the outline of an algorithm and the development of a program. The upper level sequence deals with computer methods for humanistic research. Students learn the fundamental concepts of computers and review the uses of computers in humanistic research projects. Then, non-numerical programing languages are studied in detail and students examine in depth a variety of humanistic computer applications such as the use of computers for music composition or for research in artificial intelligence. Lastly, students solve a problem using a non-numerical program. (PB)

FILMED FROM BEST AVAILABLE COPY

ED 083818

COMPUTER EDUCATION FOR THE HUMANITIES: MULTIPLE POSSIBILITIES AT THE UNIVERSITY OF SOUTH CAROLINA

Robert L. Oakman
University of South Carolina
Columbia, South Carolina 29208
(803) 777-5214

In 1967 a White House report on computers in higher education warned that "students who have not learned to use computers are badly equipped for the postgraduate world." It suggested that students of the humanities need at least a "casual" knowledge of the computer, which will be useful in some of their courses. [1] About the same time, the Department of Computer Science at the University of South Carolina, in its curriculum development, decided to implement courses utilizing the computer as an important element of humanistic education and was instrumental in hiring me in 1968 with a joint appointment in English and Computer Science. Learning to program computers is excellent training in rigorous thinking and logical reasoning, and a variety of non-numerical computer methods provide means of probing the records of human experience--in the humanities, fine arts, and social sciences--with new imagination and creative insight.

At South Carolina there are two avenues to computer education which students of the humanities may follow. To fill a lower-level quantitative requirement in the liberal arts curriculum, a freshman or sophomore student in the humanities may elect a two course sequence designed to present principles of computing as an alternative to a beginning course in mathematics or logic. For advanced undergraduates, there is a more extensive elective course emphasizing non-numerical programming and a wide range of computer applications in the humanities. In neither case is the student expected or required to have a mathematical background or interest. In fact, neither of these two courses is included in the regular curriculum for the computer science major; both are intended to appeal mainly to students of the humanities and social sciences. A closer analysis of the content and procedures of these courses will suggest the several possibilities for computer education for the humanities student at South Carolina.

The first course in the lower-level sequence, for two credits a semester, is a general "computer appreciation" course, similar to others now offered in many universities. No programming is taught; and films, exhibits, and other teaching aids are used. Topics for discussion usually include the history and development of computing machines, elements of a computer system, flowcharting, the range of computer applications, and the social impact of the computer revolution. Intended for the general student, the course has no special emphases for humanities students.

On the other hand, the second course in the sequence, entitled "Introduction to Non-numeric Processes," is specifically designed for students of the humanities or social sciences interested in non-numerical computer applications. Students of the sciences and mathematics take an alternative course emphasizing numerical applications which is the first course in the curriculum for the computer science major. Each course carries four credits per semester, with three class hours and a one credit laboratory each week. The course taken by humanists has three basic objectives: to introduce fundamental concepts of computing, to teach programming in a string-oriented computer language, and to illustrate how this language can be applied to a variety of non-numerical problems in the humanities and social sciences.

The first two weeks of the course are devoted to discussion of the five elements of a digital computer system--input, memory, processing, control, and output--and their working interrelationship. As a textbook, T. H. Crowley's Understanding Computers (New York: McGraw-Hill, 1967) is very useful for this part of the course. In a clear exposition with well chosen examples, Crowley presents the computer as a symbol-manipulating device which can be made to do many complex tasks, including non-numerical ones. Class illustrations of textbook material always stress textual information to emphasize that computers are much more than numerical machines. For example, to understand how information is stored in the internal memory of the computer, the class considers the problems of packing a sentence--as a string of binary bits representing alphabetic characters, punctuation marks, and blanks--into the core memory, with each character filling one byte of internal storage. Along with this discussion of hardware in the regular class, students are learning in the laboratory how to keypunch cards and to submit program assignments.

Eventually Crowley develops a simple hypothetical computer with a small set of machine language instructions, including the logical operations. Subsequently he discusses

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

EM 011 541

flowcharting and the creation of a computer program as a combination of initial operations. At this point, the class considers the theoretical conception of an algorithm as a step-by-step progression of operations to achieve a desired result; examples--such as flowcharting the steps of a recipe to prepare a favorite food--illustrate that algorithmic processes are found throughout human experience.

In this two week section of the course, students do their first programming with a few exercises in machine language. These assignments emphasize the logical operations of the computer rather than the numerical ones. A class exercise develops the logical operator Exclusive OR ($\oplus$) from the three logical operations--AND ($\cdot$), Inclusive OR ($+$), and NOT ($\bar{}$)--according to the well-known relation: $A \oplus B = (A \cdot \bar{B}) + (\bar{A} \cdot B)$. The Exclusive OR operation is then applied to an alphabetic problem, checking to see if two letters stored in memory are the same. The bit patterns for the two letters are combined with Exclusive OR. Only if the result is a string of zero bits are the two letters the same. The example illustrates a connection between the logical operations and a fundamental concept in non-numerical computer programming--pattern matching. The students are then assigned a final machine language program to read a string of letters, find all of the B's by using the Exclusive OR operation, and replace them with X's. The laboratory period becomes a workshop devoted to answering practical questions about flowcharting and programming.

Having been introduced to the theory and operation of computers, the students are ready for the third section of the course--learning to program in a high-level computer language suitable for analyzing textual materials. After discussing the theoretical relationship between machine language and compiler languages, the class devotes about six weeks to an examination of the basic features of a string language like SNOBOL or COMIT. Originally COMIT II was the language taught in the course because a good COMIT interpreter existed for the IBM 7040 system then in operation at the University. More recently SNOBOL4 has been taught, since the University has installed an IBM 360/65 system with a fast SNOBOL interpreter. Both languages are implemented for batch processing and include good diagnostic messages to aid in the discovery and correction of program errors. Both handle the basic operations of string manipulation--pattern matching, concatenation, deletion, duplication, and replacement--with ready facility. COMIT has more convenient list-processing techniques, but SNOBOL offers better arithmetic operations for counting or tabulation.

Discussion of the programming features of the computer language is always tied to its applicability for solving non-numerical problems of interest to students of the humanities. Class examples and a number of programs to be prepared outside of class are chosen to illustrate techniques useful in handling textual materials. For example, one outside programming assignment sets up a file of student data suitable for information retrieval. Associated with the main entry, the student's name, is a set of related information--age, sex, and job. The student's program must search the file in order to find and print the names of the twenty-five-year-old males and to count the number of female typists. Given a similarly structured file composed of publication and subject information appended to book titles, the same type of program can do automated bibliographical searching.

Several student assignments exhibit capabilities of the programming language helpful in linguistic analysis. Two problems illustrate automated syntactic analysis. Programs must negate English sentences containing any of the auxiliary verbs, such as <u>was</u> and <u>shall</u>, and must make questions out of simple declarative sentences ("Bill owns a boat" is transformed by the computer into "Does Bill own a boat?"). Perhaps the most interesting and challenging problem of this type is the creation of a program which translates English sentences into Pig Latin, with special rules for handling words beginning with vowels, <u>qu</u>, and silent consonants (as in the word <u>honor</u>). Not only do these linguistic problems offer good training in string manipulation and list processing, but they force students to examine structural characteristics of language itself which they may have never noticed.

At least two programming exercises are literary in application. A problem in computational stylistics requires the computer to dissect lines of poetry into patterns of alliteration and assonance, assigning all words with the same initial letter to the same category. There is also an assignment to alphabetize a sentence using a convenient built-in function of SNOBOL4 which orders any two strings of letters in correct lexical order. Procedures for alphabetizing are necessary in computer concordance programs and many kinds of textual data processing. With all of these problems, students of the humanities are gaining valuable practical experience in the intricacies of programming at the same time that they are learning ways that a computer can do tasks of interest to them.

Following the programming instruction, the last four weeks of the course are devoted to large-scale computer applications involving textual information and the development of algorithms for them. Topics for class discussion, which vary from term to term, may include the following: concordances, automated syntactic analysis, content analysis of historical or

political materials, and keyword-in-context indexing. Particular emphasis is given to the choice of suitable data structures for each application, so that input and string manipulation of the information will be convenient. For some applications, limited kinds of input/output equipment may require editorial decisions about how to store the data in order to retain all of its original characteristics. For instance, the necessity of distinguishing between upper and lower case letters in input and output may be of primary importance to a person preparing a computer concordance. Throughout this discussion, the problems, as well as the potentials, of computer methods for humanistic research are stressed.

Concurrent with the class discussion of applications, each student chooses a problem of interest to him and prepares a final project outside of class. He analyzes the problem, designs a data structure for input of his materials, outlines his algorithm in a flowchart, and develops a computer program to accomplish his objective. Three projects submitted one semester by students majoring in social sciences are illustrative of the kinds of non-numerical projects which have been done. A psychology major wanted to check for feminine bias in the vocabulary of grammar school reading textbooks; his project was a blend of content and stylistic analysis. He set up a dictionary of vocabulary items with masculine or feminine connotations, then read in chapters from several readers to see which words appeared most frequently. The computer searched for relevant words, printed them out, and kept a score of masculine and feminine bias. An anthropology student programmed the machine to classify pieces of Indian pottery into type and class by analyzing a series of descriptive characteristics submitted on punched cards. The co-occurrence of specific features allowed the computer to categorize the samples. A third student, studying criminology as a sociology major, designed an automated data file for the local police department. Criminal information stored for later computer retrieval included such items as type of crime, location, suspect's police record, and final disposition of the case.

These projects and the assigned programming exercises suggest the wide range of non-numerical applications that students of the humanities and social sciences are capable of preparing by the end of the course. They have gotten an introduction to fundamental principles of computing and a good dose of practical programming experience on problems relevant to their educational interests. In fact, on a class evaluation form, one student commented favorably on the combination of general background and specific instruction in a programming language:

> The fact that I learned something about computers in general and a language will help me when I begin working, for I know I will have some contact with computers. I feel I could discuss somewhat intelligently the function of a computer.

As the White House report suggested, the student expected his computer education to be useful in the future.

The alternative option at South Carolina for the humanities student is an interdisciplinary course called "Computer Methods for Humanistic Research" available for advanced undergraduate and graduate credit. Although students majoring in computer science cannot take the course for credit toward their major, students in mathematics or the sciences often elect it as a part of their computer science minors. Because of the interdisciplinary approach, students usually have widely different educational backgrounds and a variety of expectations at the start of the course. Like its lower division counterpart, the course carries four semester hours of credit (3 hours of classwork and 1 laboratory hour) and includes some of the same materials. It differs from the earlier course, however, in its heavier practical emphasis on programming and applications. In the early meetings of the laboratory, students in the humanities with no previous computer experience are rapidly introduced to fundamental concepts: elements of a computer system, input/output devices, flowcharting, keypunching.

In order to lay a common ground of understanding for all students, the first several sessions of the regular class provide an orientation to the whole field of computer research in the humanities. The class then moves directly into a close study of a non-numerical programming language. Compared to the programming instruction in the lower-level course, the emphasis here is more detailed, covers more features of the language, and is accompanied by more programming assignments. The laboratory sessions are used to reinforce programming concepts presented in class and to answer student questions. In the most recent section of the course, the language taught was SNOBOL4, but two other list-processing languages, COMIT II and LISP 1.5, and some features of FORTRAN arrays were compared to similar operations and techniques in SNOBOL.

With this detailed knowledge of a string-processing language as background, students spend the last weeks of the term--from four to six--examining a wide variety of computer applications in the humanities. The last time the course was given, presentation of this

material consisted of student reports on specific topics, as well as general class discussions of assigned library readings. Individual students reported on computer composition of music, computer collation of literary texts, and a survey of projects in artificial intelligence. Class discussions based on library materials included automated syntactic analysis, content analysis by computer, and the analysis of literary style. These discussions emphasized such important aspects of large-scale data processing as input/output methods, data structures, and programming methods.

As in the other course, each student selects a non-numerical application of interest to him and creates a computer solution for it as the final term requirement. Three student projects from the most recent offering of the course are indicative of the range of topics. One student, majoring in mathematics, set up an information data bank for the graduate English department at the University. The program queried the file of graduate student records to find out such information as the number of accumulated credits, course grades, and completion of foreign language proficiency requirements. Two linguistic projects are worthy of mention. One student programmed the machine to negate all types of French sentences--declarative, interrogative, and imperative--including the respelling of the partitive construction after negation. The other project, an application of transformational grammar, dissected a sample of English sentences into their core sentence structures; it then categorized them into three basic sentence types, depending on whether the main verb was transitive, intransitive, or a form to the auxiliary verb to be. All of these projects adapted the string capabilities of the SNOBOL language to solve different kinds of textual problems.

Being in a mixed class with humanities students seems particularly popular with the students from a scientific or mathematical background. One student's remark on the class evaluation form is typical of several others:

> I think it's a good idea to mix the two types of students because people with different backgrounds look at things differently, and much can be learned by seeing another point of view.

A girl in mathematics commented on a new, wider perspective she gained from the course: "I have been introduced to areas of computer applicability I would never have conceived of, much less studied, before."

Students of the humanities tend to emphasize other gains from the course, particularly learning to use a computer language. One undergraduate English student commented on being

> . . . particularly enthused by the fact that I could actually learn to program. Not only has the course removed all of the "hocus-pocus" and mystery that, for most humanities students, surrounds computers, but it has stimulated an interest to perhaps go on and learn other languages . . . Already I can envision how the use of COMIT and SNOBOL could help me in my future studies.

By the end of the course, all of the humanities students have been able to program with some facility. In the process, they find that they can use the computer as a new technological tool to do tasks of interest to them: stylistic, bibliographical, linguistic, or musical.

To sum up her reactions to the course, a student of linguistics commented on the broadening aspects of her experience with computers: "But to visualize its [the computer's] concept, function, and future capabilities--at least in part--will affect my thinking for years to come." She had gained insight about the theory of computing and practical experience in using a computer for her own linguistic analysis. In effect, the course had achieved its intended objectives in her case. Non-numerical applications abound today in all of the humanistic disciplines. At South Carolina we are offering multiple opportunities in the liberal arts curriculum for students like the one in linguistics to be introduced to this exciting new partnership between technology and the humanities.

REFERENCE

1. Report of the President's Science Advisory Committee, Computers in Higher Education. Washington: Government Printing Office, February, 1967, 10 and 29-30.